

Consensus in Rooted Dynamic Networks with Short-Lived Stability

Kyrill Winkler¹, Manfred Schwarz¹, and Ulrich Schmid¹

¹ TU Wien, Vienna, Austria
{kwinkler, mschwarz, s}@ecs.tuwien.ac.at

Abstract

We consider the problem of solving consensus using deterministic algorithms in a synchronous dynamic network with unreliable, directional point-to-point links, which are under the control of a message adversary. In sharp contrast to most existing work we are aware of, which has assumed oblivious message adversaries that can select the communication graph of every round arbitrarily from a set of allowed choices, we allow eventually stabilizing message adversaries, where restrictions on the sequence of the chosen graphs are allowed. This generalization permits to solve consensus also in systems with erratic boot-up phases and recovery after massive transient faults, which is impossible with oblivious message adversaries. We thoroughly address the important case of short-lived stability, where neither impossibility results nor solution algorithms were known before. We prove a tight lower bound on the minimal period of stability required for solving consensus, and provide a novel consensus algorithm that matches our lower bound, along with its correctness proof. In sharp contrast to the case of long stability periods, where standard algorithmic techniques for solving consensus can be employed, it uses quite different algorithmic techniques that avoid the detection of the stability period.

1 Introduction

We consider deterministic consensus algorithms in synchronous dynamic networks, where a potentially unknown number n of processes that never fail¹ communicate via unacknowledged messages over unreliable point-to-point links. Consensus, which is a pivotal service in truly distributed applications, is the problem of computing a common decision value based on local input values of all the processes. An execution of a consensus algorithm in our system proceeds in a sequence of lock-step synchronous² rounds, where message loss is modelled using an omniscient message adversary that determines the directed *communication graph* \mathcal{G}^r for each round r . A directed edge (p, q) present in \mathcal{G}^r means that the message sent by p in round r is successfully received by q .

In most existing work in this area, e.g. [1, 7, 16, 17], the message adversary is oblivious, i.e., may choose each \mathcal{G}^r from the *same* set of admissible graphs arbitrarily in each round. For instance, the classic result from Santoro and Widmayer [16] states that consensus is impossible iff the adversary may suppress $n - 1$ or more messages in every round. More recently, [7] introduced an equivalence relation on the set of admissible communication

¹ Nevertheless, a crash of some process p in some round could easily be modelled by p sending no messages in any later round.

² It assumes that all processes simultaneously broadcast a message at the beginning of a round, receive the messages from each other, and then simultaneously make a state transition at the end of the round, thereby proceeding to the next round. As synchronized clocks are typically required for basic communication in wireless systems anyway, e.g., for transmission scheduling and sender/receiver synchronization, this global synchrony assumption is not unrealistic: It can be implemented directly at low system levels, e.g., via IEEE 1588 network time synchronization or GPS receivers, or at higher levels via time synchronization protocols like FTSP [13].



licensed under Creative Commons License CC-BY



Leibniz International Proceedings in Informatics

LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

graphs such that consensus is solvable iff for each equivalence class there is a common *source* (a node that has a directed path to every other node) in every graph. These (and similar) approaches characterize the solvability of consensus by means of properties of the *set* of admissible graphs.

We also focus on exploring the solvability/impossibility border of consensus, albeit under *non-oblivious* message adversaries that support *eventual stabilization* [2, 3, 19]: Here, the set of admissible choices for \mathcal{G}^r may change with evolving round numbers r . Rather than constraining the set of admissible graphs, we hence constrain admissible graph *sequences*. As it turns out, consensus can be solved for graph sequences where the *set* of graphs occurring in the sequence would render consensus impossible under an oblivious message adversary [7, 16].

Apart from being theoretically interesting, considering eventually stabilizing dynamic networks is also useful from a practical perspective: Algorithms that work correctly under eventually stabilizing message adversaries are particularly suitable for systems that suffer from uncoordinated boot-up sequences or systems that must recover from massive transient faults: Network connectivity can be expected to improve over time here, e.g., due to improving clock synchronization quality. Since it is usually difficult to determine the time when such a system has reached normal operation mode, algorithms that just terminate when a reasonably stable period has been reached are obviously advantageous. Algorithms that work correctly under short-lived stable periods are particularly interesting, since they have higher coverage and terminate earlier in systems where longer stable periods occur only rarely or even not at all.

Moreover, stabilizing algorithms require less reliable and, in our case, not inherently bidirectional communication underneath, hence work with cheaper and/or more energy-efficient network communication interfaces. After all, guaranteeing reliable bidirectional communication links typically incurs significant costs and/or delays and might even be impossible in adverse environments. We hence conjecture that our findings may turn out useful for applications such as mobile ad-hoc networks [9] with heavy interference or disaster-relief applications [12].

Main contributions and paper organization: In this paper, we thoroughly answer the question of the minimal stability required for solving consensus under eventual stabilizing message adversaries. After the introduction of our system model and our message adversaries in Section 2 and Section 3, respectively, we establish the following results:

- (1) We provide a novel algorithm in Section 5, along with its correctness proof, which solves consensus for a message adversary that generates graph sequences consisting of (i) only graphs that have exactly one root component (a strongly connected component without any incoming edges from outside of the component), which (ii) contain a subsequence of $x = D + 1$ consecutive graphs with the same root component (“stable root component”). Herein, the system parameter $D \leq n - 1$ is the dynamic diameter, i.e., the number of rounds required for any node in a stable root component to reach all nodes in the system. Thanks to (i), our algorithm is always safe in the sense that agreement is never violated; (ii) is only needed to ensure termination. Compared to all existing algorithms for our non-oblivious message adversaries like [2, 19], which more or less detect the occurrence of the stability window, our algorithm uses very different algorithmic techniques.
- (2) In previous work [2], it has been shown that $x = D - 1$ is a lower bound for the stability interval for any consensus algorithm working under our message adversaries, and that

(a bound on) D must be a priori known.³ In Section 4 of this paper, we improve the lower bound to $x = D$, which reveals that the previous bound was not tight and that our algorithm is optimal. This result also shows that the mere propagation of some input value to every process does not suffice to solve consensus in our setting.

- (3) To complement earlier results [19] about consensus algorithms that work for stability periods longer than $2D + 1$, we show in Section 6 that very large⁴ periods of stability, namely, at least $3n - 3$, also allow to adopt the well-known uniform voting algorithm [6] for solving consensus in our setting.

Some conclusions and directions of future work in Section 7 complete our paper.

Related work

Research on consensus in synchronous message passing systems subject to link failures dates back at least to the seminal paper [16] by Santoro and Widmayer; generalizations have been provided in [4–7, 17]. In all these papers, consensus, resp. variants thereof, are solved in systems where, in each round, a digraph is picked from a set of possible communication graphs. The term message adversary was coined by Afek and Gafni in [1] for this abstraction.

A different approach for modeling dynamic networks has been proposed in [10]: T -interval connectivity guarantees a common subgraph in the communication graphs of every T consecutive rounds. [11] studies agreement problems in this setting. Note that solving consensus is relatively easy here, since the model assumes bidirectional and always connected communication graphs. In particular, 1-interval-connectivity, the weakest form of T -interval connectivity, corresponds to all nodes constituting a perpetually constant set of source nodes.

In both lines of research, there is no notion of eventually stabilizing behavior of dynamic networks. To the best of our knowledge, the first instance of such a message adversary has been considered in [2]: It assumed communication graphs with a non-empty set of sources and long-living periods of stability $x = 4D + 1$. [15] used message adversaries that allow a notion of “eventually forever” to establish a relation to failure detectors. Albeit we do not consider this “extremal” case in this paper, which solely addresses short-lived stability, we note that interesting insights can be drawn from this relation. [3, 20] studies consensus under a message adversary with comparably long-lived stability, which gracefully degrades to general k -set agreement in case of unfavorable conditions. However, this message adversary must also guarantee a certain influence relation between subsequently existing partitions. Finally, [19] established a characterization of uniform consensus solvability/impossibility for longer stability periods. In particular, it provides a consensus algorithm that works for stability periods of at least $2D + 1$ but does not require graph sequences where all graphs are single-rooted.

2 Model

We consider a set $\Pi = \{1, \dots, n\}$ of deterministic state machines, called *processes*, which communicate via message passing over unreliable point-to-point links. We call algorithms

³ Whereas this may seem a somewhat unrealistic (though inevitable) restriction at first sight, it must be noted that D only needs to be guaranteed throughout the stability interval. Preliminary measurements in a prototype wireless sensor network [14] showed that this is not an unrealistic assumption in this setting.

⁴ We do not consider the extreme case of eventual stability, however, where even failure-detector-based asynchronous consensus algorithm could be adopted [15].

that do not depend on n *uniform* algorithms, whereas algorithms that rely on at least some bound on n are called *non-uniform*. In this paper, we will consider exclusively non-uniform algorithms, except for Theorem 9, where we touch upon uniform algorithms as well. In any case, we assume that processes have unique ids taken from $\{1, \dots, n\}$ for simplicity; for clarity, however, we usually denote process i as p_i . Processes never fail and operate synchronously in lock-step rounds $r = 1, 2, \dots$, where each round consists of a phase of communication followed by a local state transition of every process. In the communication phase of a round, every process sends a message (possibly empty) to every other process in the system, and records the messages successfully received from the other processes. A *message adversary* (see Section 3 for the detailed definitions) is a set of graph sequences that determine which messages are lost in each round.

The *state* of a process p at the end of its round r computation is denoted by p^r , and the collection of the round r states of all processes is called round r *configuration* C^r . Those messages that are delivered by the message adversary in a given round $r > 0$ are specified via a digraph⁵ $\mathcal{G}^r = \langle \Pi, E^r \rangle$, called the round r *communication graph*. An edge $(p \rightarrow q)$ is in \mathcal{G}^r iff the round r message of p sent to q is not lost. We assume that every process p always successfully receives from itself, so the self-loops $(p \rightarrow p)$ are in every \mathcal{G}^r . The *in-neighborhood* of p in \mathcal{G}^r , $\text{In}_p(\mathcal{G}^r) = \{q \mid (q, p) \in \mathcal{G}^r\}$ hence represents the processes from which p received a message in round r .

We consider the *consensus problem*, where each process p starts with some input value x_p ; eventually, every process needs to irrevocably decide on a value (*termination*) that is the same at every process (*agreement*) and was the input of some process (*validity*). The assignment of the input values for each process is summarized in some *initial configuration* C^0 . Given a consensus algorithm \mathcal{A} , an *execution* or *run* $\varepsilon = \langle C^0, \sigma \rangle$ is hence determined by C^0 and an infinite sequence σ of communication graphs. We call an execution $\varepsilon = \langle C^0, \sigma \rangle$ *admissible* under a given message adversary MA if $\sigma \in \text{MA}$ and C^0 assigns a value to x_p for every $p \in \Pi$. Since ensuring the latter is usually trivial, we often call a sequence σ *admissible* if $\sigma \in \text{MA}$. Every message adversary is thus a set of admissible executions, which enables us to compare different message adversaries via a simple set inclusion. As in [3], we characterize a message adversary as the set of possible communication graph sequences that it may generate. A communication graph sequence σ from round a to round b is denoted as $\sigma = (\mathcal{G}^r)_{r=a}^b$, where $|\sigma| = b - a + 1$, with $b = \infty$ for infinite sequences. Applying \mathcal{A} and a finite sequence σ' to a configuration C of \mathcal{A} yields the configuration $C' = \langle C, \sigma' \rangle$ of \mathcal{A} .

As usual, we write $\varepsilon \sim_p \varepsilon'$ if the finite or infinite executions ε and ε' are *indistinguishable* to p until p decides.

Our impossibility proofs rely on a bivalence argument: Consider some algorithm \mathcal{A} that solves the binary consensus problem, where, for every process p , the initial value $x_p \in \{0, 1\}$. Given some message adversary MA, in analogy to [8], we call a configuration $C = \langle C^0, \sigma \rangle$ of \mathcal{A} *univalent* or, more specifically, *v-valent*, if all processes decided v in $\langle C, \sigma' \rangle$ for any σ' where $\sigma\sigma' \in \text{MA}$. We call C *bivalent*, if it is not univalent.

Dynamic graph concepts

First, we introduce the pivotal notion of a *root component* R , often called *root* for brevity, which denotes the vertex-set of a strongly connected component of a graph where there is no edge from a process outside of R to a process in R . Definition 1 gives its formal definition.

⁵ We sloppily write $p \in \mathcal{G}^r$ instead of $p \in V(\mathcal{G}^r)$, respectively $(p \rightarrow q) \in \mathcal{G}^r$ instead of $(p \rightarrow q) \in E(\mathcal{G}^r)$.

► **Definition 1** (Root Component). $R \neq \emptyset$ is a root (component) of graph \mathcal{G} , if it is the set of vertices of a strongly connected component \mathcal{R} of \mathcal{G} and $\forall p \in \mathcal{G}, q \in R : (p, q) \in \mathcal{G} \Rightarrow p \in R$.

It is easy to see that every graph has at least one root component. A graph \mathcal{G} that has a *single* root component is called *rooted*; its root component is denoted by $\text{root}(\mathcal{G})$. Clearly, a graph \mathcal{G} is rooted if and only if it has a rooted spanning tree: the root component is the union of the roots of all the spanning trees of \mathcal{G} . Hence, there is a directed path from every node of $\text{root}(\mathcal{G})$ to every other node of \mathcal{G} .

Conceptually, root components have already been employed for solving consensus a long time ago: The asynchronous consensus algorithm for initially dead processes introduced in the classic paper [8] relies on a suitably constructed initial clique, which is just a special case of a root component.

In order to model stability, we will rely on root components that are present in every member of a (sub)sequence of communication graphs. We call such a root component the *stable root component* of a sequence and stress that, albeit the set of processes remains the same, the interconnection topology between the processes of the root component and to the processes outside may vary greatly from round to round.

► **Definition 2** (Stable Root Component). A non-empty sequence $(\mathcal{G}^r)_{r \in I}$ of graphs has a stable root component R , iff each \mathcal{G}^r of the sequence is rooted and $\forall i, j \in I : \text{root}(\mathcal{G}^i) = \text{root}(\mathcal{G}^j) = R$. We call such a sequence a R -rooted sequence.

We would like to clarify that while “rooted” describes a graph property, “ R -rooted” describes a property of a sequence of graphs.

Given two graphs $\mathcal{G} = \langle V, E \rangle$, $\mathcal{G}' = \langle V, E' \rangle$ with the same vertex-set V , let the *compound graph* $\mathcal{G} \circ \mathcal{G}' := \langle V, E'' \rangle$ where $(p, q) \in E''$ iff for some $p' \in V : (p, p') \in E$ and $(p', q) \in E'$. Since we assume self-loops the compound graph can be written as the product of the adjacency matrices of \mathcal{G} and \mathcal{G}' .

In order to model information propagation in the network, we use a notion of *causal past*: Intuitively, a process q is in p ’s causal past, denoted $q \in \text{CP}_p^r(r')$ if $q = p$ or if, by round r , p received information (either directly or via intermediate messages) that q sent in round $r' + 1$.

► **Definition 3** (Causal past). Given an infinite sequence σ of communication graphs, the causal past of process p from (the end of) round b down to (the end of) round a is $\text{CP}_p^b(a) = \{p\}$ if $a = b$ and $\text{CP}_p^b(a) = \text{In}_p(\mathcal{G}^{a+1} \circ \dots \circ \mathcal{G}^b)$ if $a < b$.

A useful fact about the causal past is that in full-information protocols, where processes exchange their entire state history in every round, we have $q \in \text{CP}_p^r(s)$ if and only if, in round r , p knows q^s , the round s state of q .

To familiarize the reader with our notation, we present the following useful technical Lemma 4. It describes the information propagation in a graph sequence that contains an ordered set $G = \{\mathcal{G}^{r_1}, \dots, \mathcal{G}^{r_n}\}$, $i \neq j \Rightarrow r_i \neq r_j$, and $i > j \Rightarrow r_i > r_j$, of n distinct communication graphs, where any $\mathcal{G}, \mathcal{G}' \in G$ are both rooted, but $\text{root}(\mathcal{G})$ is not necessarily the same as $\text{root}(\mathcal{G}')$. As we have mentioned earlier, every $\mathcal{G} \in G$ has hence a non-empty set of sources and is therefore weakly connected. In essence, the lemma shows that, by the end of round r_n , each process p received a message from some process q that was sent after q was member of a root component of some graph of G .

► **Lemma 4.** Let $G = \{\mathcal{G}^{r_1}, \dots, \mathcal{G}^{r_n}\}$ be an ordered set of rooted communication graphs and let $X \subseteq \Pi$ with $X \cap \text{root}(\mathcal{G}^r) \neq \emptyset$ for every $\mathcal{G}^r \in G$. For every $p \in \Pi$, there is some $q \in X$ (q may depend on p) and a $\mathcal{G}^r \in G$ s.t. $q \in \text{root}(\mathcal{G}^r)$ and $q \in \text{CP}_p^{r_n}(r)$.

Proof. Let S^{r_i} be the set of those processes that, in round r_i , have a process of X in their causal past that was member of a root component so far in a graph in G . Formally, $S^{r_i} := \{p \in \Pi \mid \exists q \in X, \exists r \in \{r_1, \dots, r_i\} : q \in \text{root}(\mathcal{G}^r) \wedge q \in \text{CP}_p^{r_i}(r)\}$. In order to show the lemma, we prove by induction on i from 1 to n that $|S^{r_i}| \geq i$. We use the abbreviation $X^{r_i} := X \cap \text{root}(\mathcal{G}^{r_i})$. The base of the induction, $|S^{r_1}| \geq 1$, follows from the observation that $X^{r_1} \subseteq S^{r_1}$. For the induction step, assume for $1 \leq i < n$ that $|S^{r_i}| \geq i$. We show that then $|S^{r_{i+1}}| \geq i + 1$. If $|S^{r_i}| \geq n$, as obviously $S^{r_i} \subseteq S^{r_{i+1}}$, we are done. If $|S^{r_i}| < n$, consider that $X^{r_{i+1}} \subseteq S^{r_{i+1}}$. If $|X^{r_{i+1}} \setminus S^{r_i}| \geq 1$, we immediately have $|S^{r_{i+1}}| > |S^{r_i}|$. If $|X^{r_{i+1}} \setminus S^{r_i}| = 0$, note that there is a path in $\mathcal{G}^{r_{i+1}}$ from every $q \in X^{r_{i+1}}$ to every $p \in \Pi$. Hence, $(u, v) \in \mathcal{G}^{r_{i+1}}$ for some $u \in S^{r_i}$, $v \in \Pi \setminus S^{r_i}$, since we assumed $|S^{r_i}| < n$. As $u \in S^{r_i}$, there is some $q \in \bigcup_{r=r_1}^{r_i} X^r$ with $q \in \text{CP}_u^{r_i}(r)$. By Definition 3, since $(u, v) \in \mathcal{G}^{r_{i+1}}$, we have $q \in \text{CP}_v^{r_{i+1}}(r)$ and thus $v \in S^{r_{i+1}} \setminus S^{r_i}$ which implies $|S^{r_{i+1}}| > |S^{r_i}|$. ◀

3 Message Adversaries

First, we introduce the adversary that adheres to *dynamic diameter* D , which gives a bound on the duration of the information propagation from a stable root component to the entire network. We showed in [18, Lem. 1] that always $D \leq n - 1$; a priori restricting $D < n - 1$ also allows modelling dynamic networks where information propagation is guaranteed to be faster than in the worst case (as in expander graphs, for example).

► **Definition 5** (Dynamic diameter D). $\text{DIAM}(D)$ is the message adversary that guarantees dynamic (network) diameter D , i.e., for all graph sequences $\sigma \in \text{DIAM}(D)$, for all subsequences $(\mathcal{G}^{r_1}, \dots, \mathcal{G}^{r_{1+D-1}}) \subset \sigma$ of D consecutive R -rooted communication graphs, we have $R \subseteq \text{CP}_p^{r_1+D-1}(r_1 - 1)$ for every $p \in \Pi$.

The following liveness property, *eventual stability*, ensures that eventually every graph sequence σ has a R -rooted subsequence $\sigma' \subseteq \sigma$ of length x . Here Σ denotes the unrestricted message adversary, i.e., the set of all communication graph sequences.

► **Definition 6.** $\Diamond\text{STABILITY}(x) := \{\sigma \in \Sigma \mid \exists R \subseteq V : \sigma' \subseteq \sigma \text{ with } |\sigma'| \geq x \text{ is } R\text{-rooted}\}$.

For finite x , $\Diamond\text{STABILITY}(x)$ alone is insufficient for solving consensus: Arbitrarily long sequences of graphs that are not rooted before the stability phase occurs can fool any consensus algorithm to make wrong decisions. For this reason, we introduce a safety property in the form of the message adversary that generates only rooted graphs.

► **Definition 7.** $\text{ROOTED} := \{\sigma \in \Sigma \mid \text{every } \mathcal{G}^r \text{ of } \sigma \text{ is rooted}\}$.

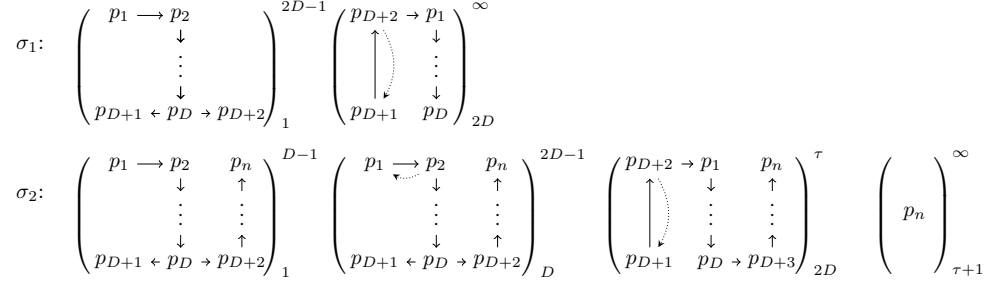
The short-lived eventually stabilizing message adversary $\Diamond\text{STABLE}_D(D+1)$ used throughout the main part of our paper adheres to the dynamic diameter D , guarantees that every \mathcal{G}^r is rooted and that every sequence has a subsequence of at least $x = D + 1$ consecutive communication graphs with a stable root component. Since processes are aware under which adversary they are, they have common knowledge of the dynamic diameter D and the duration of the stability phase x .

► **Definition 8.** We call $\Diamond\text{STABLE}_D(x) = \text{ROOTED} \cap \Diamond\text{STABILITY}(x) \cap \text{DIAM}(D)$ the short-lived eventually stabilizing message adversary with stability period x .

We observe that clearly $\Diamond\text{STABILITY}(x) \supseteq \Diamond\text{STABILITY}(D)$ for any $1 \leq x \leq D$, hence it follows that $\Diamond\text{STABLE}_D(x) \supseteq \Diamond\text{STABLE}_D(D)$.

4 Impossibility Results and Lower bounds

Even though processes know the dynamic diameter D , for very short stability periods, this is not enough to solve consensus. In Theorem 9, we prove that if processes do not have access to an upper bound on n (some N with $N \geq n$), i.e., when the algorithm is uniform, solving consensus is impossible if the period x of eventual stability is shorter than $2D$: Here, processes can never be sure whether a stable root component occurred for at least D rounds, albeit detecting such a stable root component is necessary to satisfy validity.



■ **Figure 1** Communication graph sequences of Theorem 9. A dotted edge $x \rightsquigarrow y$ represents an edge which is in \mathcal{G}^i iff it is not in \mathcal{G}^{i-1} . We assume there is an edge from every process depicted in the graph to every process not depicted in the graph.

► **Theorem 9.** *Uniform consensus is impossible under $\Diamond\text{STABLE}_D(x)$ for $0 < x < 2D$.*

Proof. As for $x > x'$, $\Diamond\text{STABILITY}(x) \subset \Diamond\text{STABILITY}(x')$, it suffices to show that consensus is impossible under message adversary $\text{MA} = \text{ROOTED} \cap \Diamond\text{STABILITY}(2D-1) \cap \text{DIAM}(D)$.

Suppose some algorithm \mathcal{A} solves consensus under MA . We provide two admissible executions $\varepsilon_1, \varepsilon_2$ (based on σ_1 , resp. σ_2 , from Figure 1) of \mathcal{A} where $\varepsilon_1 \sim_{p_{D+1}} \varepsilon_2$. We show that p_{D+1} decides 0 in ε_1 and, for almost all values of n , process p_n decides 1 in ε_2 .

Let C^0 be the initial configuration with input values $x_1 = \dots = x_{D+2} = 0$ and $x_{D+3} = \dots = x_n = 1$.

Consider execution $\varepsilon_1 = \langle C^0, \sigma_1 \rangle$ with σ_1 from Figure 1, where a dotted edge exists only in every second graph in a sequence, and all processes not depicted have an in-edge from every depicted process. $\sigma_1 \in \text{MA}$, since it guarantees eventual stability for $2D-1$ rounds, adheres to the dynamic diameter D and in every round the communication graph is rooted. By the assumed correctness of \mathcal{A} , there is a round τ by which \mathcal{A} has decided in ε_1 . The decision must be 0 because p_{D+1} only ever saw processes that knew of input value 0. This is indistinguishable for p_{D+1} from the execution where all processes did indeed start with input 0, which, according to the validity property of consensus, implies a decision on 0.

Now, consider the execution $\varepsilon_2 = \langle C^0, \sigma_2 \rangle$ with σ_2 from Figure 1. Again, $\sigma_2 \in \text{MA}$, since p_n is a stable root component for $r \geq \tau + 1$. In every round $r \leq \tau$, $p \in \{p_{D+1}, p_{D+2}\}$ have the same view in ε_1 and ε_2 : This is immediately obvious for $1 \leq r \leq D-1$. For $D \leq r \leq 2D-1$, the view of p_1 is different, but this difference is not revealed to p by the end of round $2D-1$. Finally, in rounds $2D \leq r \leq \tau$, the processes $\{p_{D+1}, p_{D+2}\}$ hear only from themselves in both executions, hence maintain $\varepsilon_1 \sim_{p_{D+1}} \varepsilon_2$.

Consequently, by round τ , p_{D+1} has decided 0. Yet, in executions where $n > \tau + D + 3$, according to ε_2 in Figure 1, we have that p_n never saw a process that had an input value different from 1. By validity and an analogous argument as above, p_n must hence decide 1 in ε_2 here, which provides the required contradiction. ◀

As our next result, we present a lower bound for the duration x of the stable period: We prove that even in the non-uniform case, consensus is impossible under $\Diamond\text{STABLE}_D(x)$ if $x \leq D$ (Theorem 11). Note that this result improves the lower bound $x \geq D - 1$ established in [2] and thus reveals that the latter was not tight.

Our lower bound can be seen as a generalization of the “lossy-link” impossibility from [17, Theorem 2], a particular formalization of consensus for two processes. There, it was shown that consensus is impossible in a two-process system where all messages except one may get lost in every round. In our terminology, this means that, for $n = 2$ and $D = 1$, consensus is impossible under $\Diamond\text{STABLE}_D(1)$, even if processes are aware of the size of the system. For general D , Theorem 11 below shows that a stability period of D or less rounds is insufficient for solving consensus for any value of N as well. Informally, the reason is that there are executions where, even with a stability phase of D rounds, some process cannot precisely determine the root component of the stability window. In Figure 2, for instance, p_{D+1} cannot determine whether p_1 alone or p_1 and p_2 together constitute the root component after a sequence of D successive occurrences of \mathcal{G}_a , respectively, \mathcal{G}_b . The determination of this root component is crucial, however, since any root component could be the “base” for a decision in the suffix of some indistinguishable execution.

Before proceeding with the proof of the main theorem, we establish an essential technical lemma. It shows that, by adding/removing a single edge at a time, we can arrive at any desired rooted communication graph when starting from any other rooted communication graph. Furthermore, during this construction, we can avoid any graphs that contain a certain “undesirable” root component R'' .

► **Lemma 10.** *Let $n > 2$, \mathcal{G} be a rooted communication graph with $\text{root}(\mathcal{G}) = \{R\}$, \mathcal{G}' be a rooted communication graph with $\text{root}(\mathcal{G}') = \{R'\}$, and R'' be some root component with $R'' \neq R$ and $R'' \neq R'$. Then, there is a sequence of communication graphs, $\mathcal{G} = \mathcal{G}_1, \dots, \mathcal{G}_k = \mathcal{G}'$ s.t. each \mathcal{G}_i of the sequence is rooted, $\text{root}(\mathcal{G}_i) \neq R''$, and, for $1 \leq i < k$, \mathcal{G}_i and \mathcal{G}_{i+1} differ only in a single edge.*

Proof. We show that for any communication graph $\bar{\mathcal{G}}$ with $\text{root}(\bar{\mathcal{G}}) = \bar{R}$, there is such a sequence $\bar{\mathcal{G}} = \bar{\mathcal{G}}_1, \dots, \bar{\mathcal{G}}_j = \bar{\mathcal{G}}'$ for any communication graph $\bar{\mathcal{G}}'$ with $\text{root}(\bar{\mathcal{G}}') = \bar{R}'$ if \bar{R}' differs from \bar{R} in at most one process, i.e., $\bar{R}' \subseteq \{\bar{R} \cup \{p\} \mid p \in \Pi\} \cup \{\bar{R} \setminus \{p\} \mid p \in \Pi\}$. Repeated application of this fact implies the lemma, because for $n > 2$ we can always find a sequence $R = R_1, \dots, R_l = R'$ of subsets of Π s.t. for each R_i of the sequence we have $R_i \neq R''$ and, for $1 \leq i < l$, R_i differs from R_{i+1} by exactly one process. To see this, we observe that in the Hasse diagram of the power set of Π , ordered by set inclusion, there are always two disjoint paths between any two subsets of Π (that do not include the empty set if $n > 2$).

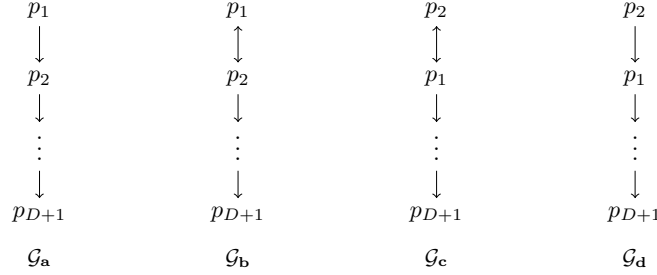
We sketch how to construct the desired communication graphs $\bar{\mathcal{G}}_i$ of the sequence in three phases.

Phase 1: Remove all edges (one by one) between nodes of \bar{R} until only a cycle (or, in general, a circuit) remains, remove all edges between nodes outside of \bar{R} until only chains going out from \bar{R} remain.

Phase 2: If we need to add a node p to $\bar{R} = \text{root}(\bar{\mathcal{G}}_i)$ to arrive at \bar{R}' , for some $q \in \bar{R}$, first add $(q \rightarrow p)$. For any $q' \neq q$ where $(q' \rightarrow p) \in \bar{\mathcal{G}}_i$ with where $p \neq q'$, remove $(q' \rightarrow p)$. Finally, add $(p \rightarrow q)$. If we need to remove a node p from $\bar{R} = \text{root}(\bar{\mathcal{G}}_i)$ to arrive at \bar{R}' , for some $(q \rightarrow p)$, $(p \rightarrow q') \in \bar{\mathcal{G}}_i$, with $q, q' \in \bar{R}$, subsequently add $(q \rightarrow q')$ and $(q' \rightarrow q)$, then remove $(p \rightarrow q)$ and $(p \rightarrow q')$.

Phase 3: Since we now already have some communication graph $\bar{\mathcal{G}}_i$ with $\text{root}(\bar{\mathcal{G}}_i) = \bar{R}'$, it

is easy to add/remove edges one by one to arrive at the topology of $\bar{\mathcal{G}}'$. First, we add edges until the nodes of \bar{R}' are completely connected among each other, the nodes not in \bar{R}' are completely connected among each other, and there is an edge from every node of \bar{R}' to each node not in \bar{R}' . Second, we remove the edges not present in $\bar{\mathcal{G}}'$. ◀



■ **Figure 2** Communication graphs for Theorem 11. We assume there is an edge from every process depicted in the graph to every process not depicted in the graph.

► **Theorem 11.** *Solving constant-offset upper-bounded non-uniform consensus is impossible under message adversary $\Diamond\text{STABLE}_D(x)$ for $1 \leq x \leq D$. This holds even if, in addition to $\Diamond\text{STABLE}_D(x)$, the adversary must guarantee that the first D rounds are R -rooted (provided the processes do not know R a priori).*

Proof. In the case where $D = 1$, we need to show the impossibility of $\Diamond\text{STABLE}_D(1)$. Note that $\sigma \in \Diamond\text{STABLE}_D(1)$ iff we have that each $\mathcal{G} \in \sigma$ is rooted and a has graph diameter of 1. Clearly, the graph sequence where (i) two fixed processes p, q have non-self-loop in-edges at most from each other and at least one of those present in every \mathcal{G}^r , and (ii) all other processes have an in-edge from both p and q is in $\Diamond\text{STABLE}_D(1)$. However, a trivial modification of [17, Theorem 2] shows that consensus (in particular, among p and q) is impossible in this setting.

For the remainder of the proof, let us thus assume $D \geq 2$. Since $\Diamond\text{STABLE}_D(x) \supset \Diamond\text{STABLE}_D(D)$ for $x \leq D$, it suffices to show the impossibility of consensus under $\Diamond\text{STABLE}_D(D)$: The execution ε where consensus cannot be solved is admissible under $\Diamond\text{STABLE}_D(x)$ if it is admissible under $\Diamond\text{STABLE}_D(D)$. The proof proceeds roughly along the lines of [17, Lemma 3]. It first shows that there is a bivalent round- D configuration for any consensus algorithm and proceeds to show by induction that every bivalent configuration has a bivalent successor configuration. Hence, any consensus algorithm permits a perpetually bivalent execution under $\Diamond\text{STABLE}_D(D)$, where consensus cannot be solved.

We show that a bivalent execution is even contained in the adversary $\Diamond\text{STABLE}'_D(D) \subseteq \Diamond\text{STABLE}_D(D)$, which consists of those executions of $\text{ROOTED} \cap \text{DIAM}(D)$ where already the first D rounds are R -rooted.

For the induction base, we show that not all round D configurations of \mathcal{A} can be univalent: Assume that some algorithm \mathcal{A} solves consensus under $\Diamond\text{STABLE}'_D(D)$ and suppose that all round D configurations of \mathcal{A} were univalent.

Let C^0 be some initial configuration of \mathcal{A} with $x_{p_1} = 0$ and $x_{p_2} = 1$ and recall the graphs $\mathcal{G}_a, \mathcal{G}_b, \mathcal{G}_c$ and \mathcal{G}_d from Figure 2. For $i \in \{a, b, c, d\}$ let $C_i^D = \langle C^0, (\mathcal{G}_i)_1^D \rangle$ denote the configuration which results from applying \mathcal{G}_i D times to C^0 . Let $\mathcal{S}(p)$ denote the star-like graph where there is an edge from the center vertex p to every other vertex and from every vertex to itself but there are no other edges in the graph. Clearly, C_a^D is 0-valent since

$\langle C_a^D, (\mathcal{S}(p_1))_{D+1}^\infty \rangle \in \Diamond \text{STABLE}'_D(D)$ and for p_1 this is indistinguishable from the situation where all processes p have $x_p = 0$. A similar argument shows that C_d^D is 1-valent.

Consider two cases:

- (1) C_b^D is 1-valent. But then C_a^D cannot be 0-valent since $\langle C_a^D, (\mathcal{S}(p_{D+1}))_{D+1}^\infty \rangle \sim_{p_{D+1}} \langle C_b^D, (\mathcal{S}(p_{D+1}))_{D+1}^\infty \rangle$.
- (2) C_b^D is 0-valent. Then C_c^D is also 0-valent since $\langle C_b^D, (\mathcal{S}(p_1))_{D+1}^\infty \rangle \sim_{p_1} \langle C_c^D, (\mathcal{S}(p_1))_{D+1}^\infty \rangle$. But then C_d^D cannot be 1-valent because $\langle C_c^D, (\mathcal{S}(p_{D+1}))_{D+1}^\infty \rangle \sim_{p_{D+1}} \langle C_d^D, (\mathcal{S}(p_{D+1}))_{D+1}^\infty \rangle$.

Hence, not all round D configurations are univalent.

For the induction step, let us assume that there exists a bivalent round r configuration C^r at the end of round $r \geq D$. For a contradiction, assume that all round $r+1$ configurations reachable from C^r are univalent. Thus, there exists a 0-valent round $r+1$ configuration $C_0^{r+1} = \langle C^r, \mathcal{G}_0 \rangle$ that results from applying some communication graph \mathcal{G}_0 to C^r . Moreover, there is a 1-valent round $r+1$ configuration $C_1^{r+1} = \langle C^r, \mathcal{G}_1 \rangle$ that results from applying some communication graph \mathcal{G}_1 to C^r .

First, let us show that for $\mathcal{G} \in \{\mathcal{G}_0, \mathcal{G}_1\}$, it holds that, if $\text{root}(\mathcal{G}) = \text{root}(\mathcal{G}^r)$, there is an applicable graph \mathcal{G}' s.t. $\langle C^r, \mathcal{G}' \rangle$ has the same valency as $\langle C^r, \mathcal{G} \rangle$ and $\text{root}(\mathcal{G}) \neq \text{root}(\mathcal{G}')$. The reason for this is that we can construct \mathcal{G}' from \mathcal{G} by simply adding an edge (p, q) for some $q \neq p \notin \text{root}(\mathcal{G})$, $q \in \text{root}(\mathcal{G})$ if $|\text{root}(\mathcal{G})| = 1$, respectively, by removing (p, q) for some $q \in \text{root}(\mathcal{G})$ and all $p \neq q$ if $|\text{root}(\mathcal{G})| > 1$. This yields a graph \mathcal{G}' with the desired property because $\langle C^r, \mathcal{G}, (\mathcal{S}(p))_{r+1}^\infty \rangle \sim_p \langle C^r, \mathcal{G}', (\mathcal{S}(p))_{r+1}^\infty \rangle$. The applicability of \mathcal{G}' follows because it is rooted by construction and $\text{root}(\mathcal{G}') \neq \text{root}(\mathcal{G}^r)$.

This means that we can find graphs $\mathcal{G}'_0, \mathcal{G}'_1$ s.t. $\text{root}(\mathcal{G}'_0) \neq \text{root}(\mathcal{G}^r)$, $\text{root}(\mathcal{G}'_1) \neq \text{root}(\mathcal{G}^r)$, and $\langle C^r, \mathcal{G}'_0 \rangle$ is 0-valent while $\langle C^r, \mathcal{G}'_1 \rangle$ is 1-valent. As we assumed $D \geq 2$ it follows that $n > 2$. We can hence apply Lemma 10 to go from \mathcal{G}'_0 to \mathcal{G}'_1 by adding/removing a single edge at a time, without ever arriving at a graph that has more than one root component or has the same root component as \mathcal{G}^r . Somewhere during adding/removing a single edge, we transition from a graph \mathcal{G}_i to a graph \mathcal{G}_{i+1} , by modifying an edge (p, q) , where the valency of $C = \langle C^r, \mathcal{G}_i \rangle$ differs from the valency of $C' = \langle C^r, \mathcal{G}_{i+1} \rangle$. Nevertheless, \mathcal{G}_i and \mathcal{G}_{i+1} , are applicable to C^r because they are rooted and have a different root component as \mathcal{G}^r , hence guarantee the membership of the sequence in $\text{DIAM}(D)$ for $D > 1$. However, C and C' cannot have a different valency because $\langle C, (\mathcal{S}(p))_{r+1}^\infty \rangle \sim_p \langle C', (\mathcal{S}(p))_{r+1}^\infty \rangle$. This is a contradiction and hence not all round $r+1$ configurations can be univalent. ◀

5 Solving Consensus with $D + 1$ Rounds of Stability

We now present Algorithm 1, which, under the message adversary $\Diamond \text{STABLE}_D(D+1)$, solves consensus if D and a bound $N \geq n$ is known a priori. It relies on an underlying graph approximation algorithm used already in [3, 18], which provides each process with a local estimate of past communication graphs. It is easily implemented on top of any full-information protocol⁶ and has been omitted for brevity. As stated in Corollaries 12 and 13 below, the local graph estimates allow to faithfully detect the existence of root components with a latency of at most D rounds, with some restrictions.

⁶ Since we are mainly interested in the solvability aspect of consensus, we consider a full-information protocol where processes forward their entire state in every round. As will become apparent from the correctness proof of Algorithm 1, however, it would in fact suffice to exchange the relevant data structures for the last $N(D + 2N)$ rounds at most. Hence, the same engineering improvements as outlined in [18] can be used to get rid of the simplifying full-information-protocol assumption.

In our consensus algorithm, processes operate in two alternating states (“not locked” and “locked”). Basically, in the “not locked” state, process p tries to find a root component and, if successful, locks on to it, i.e., considers it a potential base for its decision. This is realized by adapting the local prop_p value to the value of the root component (which is the maximum of the proposal values of its members) and setting $\text{locked}_p \leftarrow \text{TRUE}$, thereby entering the “locked” state. Subsequently, p waits for contradictory evidence for a period of time that is long enough to guarantee that every process in the system has adapted to the value of the root component locked-on by p . If p finds contradictory evidence, it backs off by leaving the “locked” state through setting $\text{locked}_p \leftarrow \text{FALSE}$.

In more detail, p changes its state from “not locked” to “locked” when it detects a root component via Line 10 that was present D rounds before the current round. In this case, p enters the locked state by setting $\text{locked}_p \leftarrow \text{TRUE}$ and prop_p to prop' , the maximum value known to any process of the root component in round $r - D$, via Lines 13 and 14. It furthermore records the current round as ℓ , the so-called *lock-round*, via Line 15. This is an optimistic mechanism, i.e., the process “hopes” that this root component was already the stable root component promised by the adversary. In this sense, the process starts collecting evidence, possibly contradicting its hope, that stems from the lock-round itself or a more recent round.

If p finds a root component when it is already in the “locked” state, i.e., when it is already locked-on to some root component, it proceeds as follows: First, p checks whether it detected a relevant change in the members of the root component since its lock-round and whether the maximum value, known to the processes of the new root component at the time, is different from its current proposal value. If both checks succeed, there was enough instability for the process to conclude that the currently locked-on root component cannot belong to the stable period; it hence locks-on to the newly detected root component. If the values remained the same, it puts the current round in a *queue* of candidate lock-rounds in Line 16. Later, if the process finds evidence that indeed contradicts its hope (as detailed below), it will pick the next lock round from the candidate queue via Line 19, and tries to find contradictory evidence from this round on, thereby ensuring that it cannot miss the promised stable root component.

When p is still in a “locked” state after completing its root component detection, it searches for contradictory evidence. More precisely, contradictory evidence means that p learned of another process q s.t. q is not locked-on to a root component with the same value as p . This is realized via the guard of Line 17. If p finds such evidence, it leaves its locked state and enters the “not locked” state again. A decision occurs when p has received no contradictory evidence for such a long time that every other process has seen that p might be in this situation via Line 20. As we prove in Lemma 16 below, it is guaranteed that if p passes the guard in Line 20, then every other process has p ’s decision value as its local proposal value forever after. Hence, all future decisions will be based on this value, which leads the system to a safe configuration.

Correctness proof

We now prove the correctness of Algorithm 1 under $\Diamond\text{STABLE}_D(D+1)$. As shown in [18, Lemmas 3 and 4], the simple graph approximation algorithm running underneath our consensus algorithm allows processes to faithfully detect root components under certain circumstances. Denoting process p ’s round r estimate of $\text{root}(\mathcal{G}^s)$ as $\text{root}_p^r(s)$, the following two key features can be guaranteed algorithmically (we assume that $\text{root}_p^r(s)$ returns $\{\perp\}$ if p is unsure about its estimate):

Algorithm 1: Consensus algorithm, code for process p

Let r_q^* be the last round p heard from q in round r , i.e., $q \in \text{CP}_p^r(r_q^*)$ and $q \notin \text{CP}_p^r(r_q^* + 1)$.

Initialization:

```

1   $\text{prop} \leftarrow x_p$  /* initially,  $p$  proposes own input */
2   $\text{locked} \leftarrow \text{TRUE}$  /*  $p$  starts 'locked-on' */
3   $\ell \leftarrow 1$  /* initialize lock-round to start round */
4   $\text{queue} \leftarrow \emptyset$  /* we assume that  $\max(\emptyset) = 0$  */

Round  $r$  computation:
5   $R \leftarrow \text{root}_p^r(r - D)$ 
   /* Find all roots that were detected since the (estimated) start of the
   stability phase */
6   $T \leftarrow \{\text{root}_p^r(i) \mid \max(\max(\text{queue}), \ell) - D \leq i \leq r - D\}$ 
   /* For each process, determine its known states within the last  $N$  rounds: */
7   $S \leftarrow \{q^s \mid q \in \text{CP}_p^r(r - N) \wedge s \in [r - N, r_q^*]\}$ 
   /* Find the proposal values of the locked-on processes of  $S$  */
8   $S' \leftarrow \{\text{prop}_q^s \mid q^s \in S \wedge \text{locked}_q^s = \text{TRUE}\}$ 
   /* Collect all known states of the last  $N(D + 2N)$  rounds */
9   $S'' \leftarrow \{q^s \mid q \in \text{CP}_p^r(r - N(D + 2N)) \wedge s \in [r - (D + 2N)^2, r_q^*]\}$ 
10 if  $R \neq \{\perp\}$  then
11    $\text{prop}' \leftarrow \max_{q \in R} \text{prop}_q^{r-D}$ 
12   if  $\text{locked} = \text{FALSE}$  or  $(\text{prop}' \neq \text{prop} \text{ and } R', R'' \in T \text{ with } R' \neq R'')$  then
13      $\text{prop} \leftarrow \text{prop}'$ 
14      $\text{locked} \leftarrow \text{TRUE}$ 
15      $\ell \leftarrow r$ 
16   else if  $\text{prop}' = \text{prop}$  then add  $r$  to queue
17 if  $r \geq \ell + N$  and  $q^s \in S$  s.t.  $\text{locked}_q^s = \text{FALSE}$  or  $\text{prop} \neq \text{prop}_q^s$  then
18   remove all  $r'$  with  $r' \leq s$  from queue
19   if  $\text{queue} \neq \emptyset$  then  $\ell \leftarrow \min(\text{queue})$  else  $\text{locked} \leftarrow \text{FALSE}$ 
20 if  $r \geq \ell + 2N$  and  $S'$  contains single element  $x \neq \text{prop}$  then  $\text{prop} \leftarrow x$  if not yet decided,
    $r = \ell + N(D + 2N)$ , and, for all  $q^s$  of  $S''$ ,  $\text{locked}_q^s = \text{TRUE}$  and  $\text{prop}_q^s = \text{prop}$  then decide
   prop

```

► **Corollary 12.** If $\text{root}_p^r(s) \neq \{\perp\}$, then $\text{root}(\mathcal{G}^s) = \text{root}_p^r(s)$. Furthermore, in round r , process p knows the round s state q^s for every $q \in \text{root}(\mathcal{G}^s)$.

Note that Corollary 12 relies on the fact that the graph approximation algorithm maintains an under-approximation of past communication graphs.

► **Corollary 13.** If $(\mathcal{G}^r)_{r=s}^{s+D}$ is R -rooted, then we have $\text{root}_p^{s+D}(s) = R$ for every process p .

Note that Corollary 13 is a consequence of the message adversary respecting the dynamic diameter D : In round $s + D$, for each process $q \in R$, every process received a message containing q^s , the round s state of q .

We are now ready to formally analyze Algorithm 1. For this purpose, we introduce the useful term of *v-locked root component* to denote a root component where all members are *v-locked* for the same v , i.e., have the proposal v and are in the locked state.

► **Definition 14.** The round r state p^r of p is a *v-locked* state if $\text{locked}_p^r = \text{TRUE} \wedge \text{prop}_p^r = v$. $R = \text{root}(\mathcal{G}^r)$ is a *v-locked* root component if all q^r with $q \in R$ are *v-locked*.

The following technical lemma is key for the proof of the agreement property of consensus. It assures that if a sequence of at least $D + 2N$ communication graphs occurs in which all

root components happen to be v -locked by our algorithm, then all the processes' proposal values are v in all subsequent rounds.

► **Lemma 15.** *Let $\sigma = (\mathcal{G}^r)_{r=a}^c$ be a sequence of communication graphs such that, for some fixed value v , every $\mathcal{G}^r \in \sigma$ has a v -locked root component and $|\sigma| \geq D + 2N$. Then, for any process p and all rounds $r \geq c$, we have $\text{prop}_p^r = v$.*

Proof. Let $b = a + D + N - 1$ and note that $c \geq a + D + 2N - 1 = b + N$. We prove our lemma using the invariants $\mathcal{I}, \mathcal{I}'$ that are defined as follows: $\mathcal{I}_p(r) \Leftrightarrow \text{locked}_p^r = \text{FALSE} \vee \text{prop}_p^r = v$, $\mathcal{I}'_p(r) \Leftrightarrow \text{prop}_p^r = v$. We use $\mathcal{I}(r)$, resp. $\mathcal{I}'(r)$, to denote that $\mathcal{I}_p(r)$, resp. $\mathcal{I}'_p(r)$, holds for every $p \in \Pi$.

First, we show $\mathcal{I}(r)$ for $r \in [b, c]$, by proving that for any process p , $\mathcal{I}_p(r)$ holds. We distinguish two cases.

(1) If p 's lockround ℓ , just before executing Line 17 in round r , satisfies $\ell > r - N$, we have that $\text{prop}_p^\ell = v$, because of Corollary 12 and because $\text{root}(\mathcal{G}^{\ell-D})$ is v -locked. If p modified prop_p in some round $s \in [\ell, r]$, it must have done so via Line 13 or Line 20. The former again means that $\text{prop}_p^s = v$, due to Corollary 12 and since $\text{root}(\mathcal{G}^{s-D})$ is v -locked by assumption. The latter means that the guard of Line 20 was passed and hence $s \geq \ell + 2N > b + N$, which contradicts that $s \in [\ell, r] \subseteq [b - N + 1, c]$.

(2) If $\ell \leq r - N$, i.e., $r \geq \ell + N$, because we assume that $\text{root}(\mathcal{G}^i)$ is v -locked for each \mathcal{G}^i of $(\mathcal{G}^i)_{i=a}^r$ and $r \in [b, c] \Rightarrow r \geq a + N$, it follows from Lemma 4 that there is a process q and a round $s \in [r - N, r]$ s.t. $q \in \text{root}(\mathcal{G}^s)$, $q \in \text{CP}_p^r(s)$, and q^s is v -locked. Since $\text{root}(\mathcal{G}^s)$ is v -locked by assumption, S contains some state q^s with $\text{prop}_q^s = v$ and $\text{locked}_q^s = \text{TRUE}$. Therefore, $v \in S^r$. It follows that if S^r contains a single element x , we have $x = v$. Since Line 20 is the only place in the code, apart from Line 13, where p assigns a value to prop_p^r , if $\text{prop}_p^{r-1} = v$, then $\text{prop}_p^r = v$. On the other hand, if $\text{prop}_p^{r-1} \neq v$, because $r \geq \ell + N$, p evaluates the guard of Line 17 to true. If $\text{queue} = \emptyset$, p executes Line 19 and sets $\text{locked}_p^r \leftarrow \text{FALSE}$. If $\text{queue} \neq \emptyset$, let $\ell' = \min(\text{queue})$. Due to Line 18, we have $\ell' > s$, and because $s \in [r - N, r]$, it follows that $\ell' > r - N > b - N > a + D - 1$. Hence $\ell' - D \in [a, c - D] \subseteq [a, c]$. This means that during round ℓ' , directly after executing Line 11, p already had $\text{locked} = \text{TRUE} \wedge \text{prop} = \text{prop}'$. But according to Corollary 12, this implies that $\text{prop} = v$, because $\text{root}(\mathcal{G}^{\ell'-D})$ is v -locked by assumption during the interval $[a, c]$.

We are now ready to complete the proof by using induction to show that $\mathcal{I}'(r)$ holds for each $r \geq c$. For the base case, we prove $\mathcal{I}'_p(c)$ for an arbitrary process p :

If the lockround ℓ of p in round c , just before reaching line 15, satisfies $\ell > c - 2N$, we can use the same arguments as in (1) above: It follows from the assumption that $\text{root}(\mathcal{G}^i)$ for $i \in [c - 2N - D + 1, c] \subseteq [a, c]$ is v -locked and Corollary 12 that whenever p changes its proposal value in some round s , we have $\text{prop}_p^s = v$. Consequently, $\text{prop}_p^c = v$ as well.

If $\ell \leq c - 2N$, we are in a similar situation as in (2) above as $v \in S_p^c$. Since $\mathcal{I}_p(s)$ holds for $s \in [b, c]$, $\text{locked}_q^s = \text{TRUE}$ implies that $\text{prop}_q^s = v$. Hence, there can be no $x \neq v$ with $x \in S_p^c$. Therefore, p executes Line 20 in round c and sets $\text{prop} \leftarrow v$, hence $\text{prop}_p^c = v$.

For the induction step, we show that, for any process p , $\bigwedge_{i=c}^r \mathcal{I}'(i) \Rightarrow \mathcal{I}'_p(r+1)$. We distinguish three cases:

(1) p assigns a value to prop in round $r+1$ in Line 13. Irrespectively of r , we have $\text{prop}_p^{r+1} = v$: If $r < c + D$, this holds because the root of \mathcal{G}^{r+1-D} is v -locked. If $r \geq c + D$, we get the result from the fact that $\mathcal{I}'(r-D)$ holds.

(2) p assigns a value to prop in round $r+1$ in Line 20. This means that S' contains only a single value. Clearly, for arbitrary rounds r' , $\mathcal{I}'(r') \Rightarrow \mathcal{I}(r')$. Therefore, if S_p^{r+1} contains prop_q^s for $s \geq r+1 - N \geq b$, because $\text{locked}_q^s = \text{TRUE}$ and $\mathcal{I}(s)$ holds, we have $\text{prop}_q^s = v$.

(3) p does not assign a new value to prop_p^{r+1} . Then, $\text{prop}_p^{r+1} = v$, by the hypothesis that $\mathcal{I}'_p(r)$ holds. \blacktriangleleft

With these preparations, we can now prove agreement, validity and termination of our consensus algorithm.

► **Lemma 16.** *Algorithm 1 ensures agreement under each sequence $\sigma \in \Diamond\text{STABLE}_D(D+1)$.*

Proof. We show that if a process p decides v in round r , all future decisions are v as well. A decision v by p can only occur if p executed Line 20, which means that $S_p^{'''r}$ contains only states q^s with $\text{locked}_q^s = \text{TRUE}$ and $\text{prop}_q^s = v$. We show below that this implies that there was a sequence $\sigma'' \subseteq \sigma' = (\mathcal{G}^i)_{i=r-N(D+2N)}^r \subset \sigma$ s.t. each $\mathcal{G}^i \in \sigma''$ has a v -locked root component and $|\sigma''| \geq D + 2N$. We can hence directly apply Lemma 15, which yields that $\text{prop}_q^t = v$ for all $q \in \Pi, t \geq r$. This proves our claim because a decision at process q in round t can only be on prop_q^t .

We show the contrapositive of the above implication. Assume there is no sequence $\sigma'' \subseteq \sigma'$ with a common v -locked root component and $|\sigma''| \geq D + 2N$. Thus, every subsequence $\sigma''' \subseteq \sigma'$ that has a common v -locked root component may have at most $|\sigma'''| < D + 2N$ and is followed immediately by a communication graph \mathcal{G}^{r_i} where some process $q \in \text{root}(\mathcal{G}^{r_i})$ has $\text{locked}_q^{r_i} = \text{FALSE}$ or $\text{prop}_q^{r_i} \neq v$. Hence, we have a set G of at least n such communication graphs during σ' . More accurately, there is an ordered set of communication graphs $G = \{\mathcal{G}^{r_1}, \dots, \mathcal{G}^{r_n}\}$ with $r_1 \geq r - N(D + N)$, $r_n \leq r$, and some set $X \subseteq \Pi$ of processes such that $X \cap \text{root}(\mathcal{G}^{r_i}) \neq \emptyset$ for every $\mathcal{G}^{r_i} \in G$. Thus $\exists q \in X \cap \text{root}(\mathcal{G}^{r_i})$ s.t. $\text{locked}_q^{r_i} = \text{FALSE}$ or $\text{prop}_q^{r_i} \neq v$. By Lemma 4, there is hence some process $q \in R$ and a $\mathcal{G}^{r_i} \in G$ s.t. $q \in \text{root}(\mathcal{G}^{r_i})$ and $q \in \text{CP}_p^{r_n}(r_1)$ where $r_1 \leq r_i \leq r_n$. Consequently, $q^{r_i} \in S_p^{'''r}$, but then there is a state q^{r_i} in $S_p^{'''r}$ with $\text{locked}_q^{r_i} = \text{FALSE}$ or $\text{prop}_q^{r_i} \neq v$. \blacktriangleleft

► **Lemma 17.** *Algorithm 1 ensures validity.*

Proof. Since decisions are only on values of prop_p^r , which are modified exclusively by either assigning prop_q^s from different processes or input values x_i , validity follows. \blacktriangleleft

► **Lemma 18.** *Algorithm 1 eventually terminates under any sequence σ of $\Diamond\text{STABLE}_D(D+1)$.*

Proof. Since $\sigma \in \Diamond\text{STABLE}_D(D+1)$, there is an earliest subsequence $\sigma' = (\mathcal{G}^r)_{r=a}^b \subset \sigma$ with $|\sigma'| = D + 1$ that has a common root R . Let $v = \max_{q \in R} \text{prop}_q^a$.

We prove that in any round $r \geq b$, all processes are v -locked and have $\ell^b \leq b$. This implies the lemma, because, by Line 9, in round $b + N(D + 2N)$, $S^{''b+N(D+2N)}$ contains only v -locked states. Hence, any process that has not decided yet in round $b + N(D + 2N)$ will execute Line 20 and decide v .

In order to prove this, we introduce the invariants $\mathcal{J}_p(r) \Leftrightarrow \text{locked}_p^r = \text{TRUE} \wedge \text{prop}_p^r = v \wedge (\ell_p^r = b \vee b \in \text{queue}_p^r) \wedge \ell_p^r \leq b$ and $\mathcal{J}(r) \Leftrightarrow \forall p \in \Pi: \mathcal{J}_p(r)$, which state that, at the end of round r , p , resp. every process of Π , is v -locked and either p 's lock-round is b or b is queued at p . We use induction to show that for all $r \geq b$ it holds that $\bigwedge_{i=b}^r \mathcal{J}(i) \Rightarrow \mathcal{J}(r+1)$.

For the base case, we show that $\mathcal{J}_p(b)$ for any $p \in \Pi$. According to Corollary 13, in round b , every process p has $\text{root}_p^b(a) = R$ and the guard of Line 10 is evaluated to true. We distinguish two cases:

(1) p is not v -locked at the beginning of its round b computation, i.e., $\text{locked}_p^{b-1} = \text{FALSE}$ or $\text{prop}_p^{b-1} \neq v$. If $\text{locked}_p^{b-1} = \text{FALSE}$, clearly Lines 13, 14, and 15 are executed. If $\text{prop}_p^{b-1} \neq v$, because σ' is the first subsequence with the desired properties by assumption,

$\text{root}(\mathcal{G}^{a-1}) \neq R$. According to Corollary 12, $R' = \text{root}_p^b(a-1) \neq R$, thus, by construction of T , we have $R', R \in T$ with $R' \neq R$. Since we assumed that $\text{prop}_p^{b-1} \neq v = \text{prop}_p'$, also in this subcase, Lines 13, 14, and 15 are executed. Before executing Line 17, we hence have $\ell_p = b$ and therefore none of the subsequent guards can be triggered. Consequently, we have that p is still v -locked at the end of its round b computation and $\ell_p^b = b$.

(2) If p is already v -locked at the end of round $b-1$, we have $\ell_p^{b-1} < b$ and p executes Line 16 in round b , thereby adding b to queue_p . Since ℓ is not modified later, $\ell_p^b < b$. If p does not enter the guard of Line 17, $\mathcal{J}_p(b)$ holds before p executes Line 20. Otherwise, queue_p still contains b . Since now only states q^s with $s < b$ are known to p , Line 18 cannot remove b from queue_p . Hence, the guard of Line 19 is true and $\mathcal{J}_p(b)$ still holds before p executes Line 20.

Since we assume that p was already v -locked at the end of round r , this continues to hold also in the case when Line 20 is ever executed: Since $v \in S'$, it must be the case that if S' indeed contains a single value x , then $x = v$. Therefore, $\mathcal{J}_p(b)$ holds at the end of round b as asserted.

For the induction step, we assume that $\bigwedge_{i=b}^r \mathcal{J}(r)$ is true for some $r \geq b$ and show that then $\mathcal{J}_p(r+1)$ holds for an arbitrary process p .

We first show that, if p enters Line 10, it cannot pass the conditional of Line 12. For this, let us distinguish two cases:

(1) For $r \in [b, b+D]$, $\mathcal{J}(r)$ implies that either $b = \ell_p^r$ or $b \in \text{queue}_p^r$ and p is v -locked at the end of round r . Thus, in round $r+1$, Line 12 can only be passed if $R', R'' \in T$ s.t. $R' \neq R''$. Since obviously $\max(\text{queue}) < r$ and we assumed that $r \in [b, b+D]$, by the construction of T in Line 6, $T \subseteq \{\text{root}_p^r(i) \mid a \leq i \leq b\}$. In fact, $T \subseteq \{\text{root}(\mathcal{G}^i) \mid a \leq i \leq b\} \cup \{\perp\}$, as a consequence of Corollary 13. By the assumption that σ' is R -rooted, after its construction in round r , $T = \{R\}$ and Line 12 cannot be entered.

(2) For $r \geq b+D+1$, by the induction hypothesis $\mathcal{J}(r+1-D)$ holds. Thus, $\text{locked}_p^r = \text{TRUE}$ and for any process q , $\text{prop}_q^{r-D} = v$, which implies $\text{prop}_p' = v$. Therefore, the guard of Line 12 cannot be passed.

Note that, by the hypothesis $\mathcal{J}_p(r)$, $\ell_p^{r+1} \leq b$, since Line 12 was not passed and ℓ is not modified after this point.

Until now we have shown that $\mathcal{J}_p(r+1)$ holds right before p executes line Line 17. We again distinguish two cases:

(1) If $\ell \geq b$, it follows from the induction hypothesis $\bigwedge_{i=b}^r \mathcal{J}(i)$ that all states from round ℓ to round r are v -locked and hence S cannot contain any state that is not v -locked. Consequently, p cannot pass the guard of Line 17.

(2) If, on the other hand, $\ell < b$, the hypothesis $\mathcal{J}(r)$ implies that $b \in \text{queue}_p$. Therefore, if the guard of Line 17 is true, Line 19 is executed and Line 19 cannot be entered.

In both cases $\mathcal{J}_p(r+1)$ still holds before p reaches Line 20.

Observe that, according to the induction hypothesis $\mathcal{J}(r)$, p was v -locked at the end of round r . Hence, $v \in S'$, and if S' contains a single element x , it must be that $x = v$. Hence $\mathcal{J}_p(r+1)$ also holds if p executed Line 20. Since Line 20 cannot not invalidate $\mathcal{J}_p(r+1)$ either, we are done. \blacktriangleleft

Lemmas 16, 17, and 18 yield the correctness of Algorithm 1:

► **Theorem 19.** *Algorithm 1 correctly solves consensus under $\diamond\text{STABLE}_D(D+1)$, provided $N \geq n$.*

6 Solving Consensus for Long Periods of Eventual Stability

While the ability to solve consensus under short-lived periods of stability is beneficial in terms of assumption coverage and fast termination, it comes at the price of the rather involved Algorithm 1. In this section, we show that a (considerably) longer period of stability facilitates a (considerably) simpler algorithm. More specifically, we show that, for $\Diamond\text{STABLE}_D(x)$ with $x \geq 3n - 3$, a simple consensus algorithm can be obtained by adopting existing algorithmic techniques [6]. Unfortunately, though, this approach does not work for $x < 3n - 3$.

With Σ representing all possible communication graph sequences, let **NON-SPLIT** be defined as those $\sigma \in \Sigma$ for which the following holds: If $\mathcal{G}^r \in \sigma$, then for all $p, q \in \mathcal{G}^r$, there is some $q' \in \mathcal{G}^r$ such that $(q', p) \in \mathcal{G}^r$ and $(q', q) \in \mathcal{G}^r$. Furthermore, let $\Diamond\text{UNIFORM}$ represent those sequences where there is a round r and a set P of processes s.t. $\mathcal{G}^r = \bigcup_{p \in P} \mathcal{S}'(p)$, with $\mathcal{S}'(p)$ denoting the star graph with central node p and no self-loops in the non-center node. Finally, let $\Diamond\text{STAR}(y)$ be the set of those $\sigma \in \Sigma$ that contain a subsequence $\sigma' \subseteq \sigma$, $|\sigma'| \geq y$, with a stable root component R satisfying $\bigcup_{p \in R} \mathcal{S}'(p) \subseteq \mathcal{G}^r$ for each $\mathcal{G}^r \in \sigma'$.

Note that the definition of $\Diamond\text{UNIFORM}$ is such that if $(q, q) \in \mathcal{G}^r$ then $q \in P$, i.e., processes outside of P cannot have any outgoing edges (including self-loops). This restriction does not hold for $\Diamond\text{STAR}(y)$, however, which is hence a strictly stronger adversary for $y = 1$, i.e., $\Diamond\text{STAR}(1) \supset \Diamond\text{UNIFORM}$.

Given a sequence $\sigma = (\mathcal{G}^i)_{i \geq 0}$, let the compound sequence $\tilde{\sigma} = (\tilde{\mathcal{G}}^i)_{i \geq 0}$ be the sequence of graphs $\tilde{\mathcal{G}}^r = \mathcal{G}^{(r-1)(n-1)+1} \circ \dots \circ \mathcal{G}^{r(n-1)}$. We recall from [5, Lemma 4] that if $\sigma \in \text{ROOTED}$ then $\tilde{\sigma} \in \text{NON-SPLIT}$.

Since the compound graphs $\tilde{\mathcal{G}}^r$ combine n consecutive graphs \mathcal{G}^r in fixed (ascending) order, this construction is not necessarily aligned with the R -rooted subsequence provided by $\Diamond\text{STABLE}_D(y)$. Let $\tilde{\sigma}$ be the compound sequence of some $\sigma \in \Diamond\text{STABLE}_D(y)$. In the worst case, $\tilde{\sigma}$ may contain a sequence of only $\lfloor \frac{y-n+1}{n-1} \rfloor$ compound graphs $\tilde{\mathcal{G}}^r$ with the property $\bigcup_{p \in R} \mathcal{S}'(p) \subseteq \tilde{\mathcal{G}}^r$, i.e., $\tilde{\sigma} \in \Diamond\text{STAR}(\lfloor \frac{y-n+1}{n-1} \rfloor)$.

From [6, Theorem 5], we know that consensus can be solved under $\text{NON-SPLIT} \cap \Diamond\text{UNIFORM}$. On the other hand, our impossibility result Theorem 11 (for $D = n - 1$), implies that consensus is impossible under $\text{NON-SPLIT} \cap \Diamond\text{STAR}(1)$. Note carefully that we can allow the R -rooted subsequence to be aligned with the compound graph construction here, hence the impossibility for $\Diamond\text{STAR}(1)$.

Consequently, for $i < 3$, using compound graphs does not provide a solution for the message adversary $\text{ROOTED} \cap \Diamond\text{STABILITY}(i(n-1))$, since $\sigma \in \text{ROOTED} \cap \Diamond\text{STABILITY}(i(n-1))$ implies only $\tilde{\sigma} \in \Diamond\text{STAR}(i-1) \cap \text{NON-SPLIT}$ according to the considerations above. For $i \geq 3$, however, $\tilde{\sigma} \in \Diamond\text{STAR}(2) \cap \text{NON-SPLIT}$, which allows us to derive a simple consensus algorithm Algorithm 2 by combining the detection of root components with the uniform voting algorithm from [6, Algorithm 5].

► **Theorem 20.** *Algorithm 2 solves Consensus under $\Diamond\text{STAR}(2) \cap \text{NON-SPLIT}$.*

Proof. If process q is the first to decide v , at the end of round r , then every process has $\text{prop}^r = v$: Since q received only messages $(v, *)$ with $v \neq \perp$ and \mathcal{G}^r is non-split, p received a message $(v, *)$. Moreover no message with $(v', *)$ and $v' \neq v$ was sent in round r , because every message with some $m \neq \perp$ was generated based on the root of round $r - 1$ which is, if detected by a process, detected consistently on every process. Hence during the round r computation, every process sets prop to v either via line 13 or line 18. As all future decisions are based on v , this ensures agreement.

Algorithm 2: Compound graph algorithm, code for process p

Initialization:

- 1 $\text{prop} \leftarrow x_p$
- 2 $m \leftarrow \perp$

Transmit round r messages:

- 3 Send (m, prop) to all q with $(p, q) \in \mathcal{G}^r$
- 4 Receive (m_q, prop_q) from all q with $(q, p) \in \mathcal{G}^r$

Perform round r computation until decision:

- 5 $M \leftarrow \bigcup_{(q,p) \in \mathcal{G}^r} (q, m_q, \text{prop}_q)$
- 6 $R^{r-1} \leftarrow \text{root}_p^r(r-1)$
- 7 **if** $\bigcup_{(q,m_q,\text{prop}_q) \in M} m_q = \{v\}$ **with** $v \neq \perp$ **then**
 - 8 $\text{prop} \leftarrow v$
 - 9 $m \leftarrow v$
 - 10 decide v
- 11 **else if** $R^{r-1} \neq \emptyset$ **then**
 - 12 **if** $q \in R^{r-1}$ **with** $(q, m_q, \text{prop}_q) \in M$ **s.t.** $m_q \neq \perp$ **then**
 - 13 $\text{prop} \leftarrow m_q$
 - 14 **else**
 - 15 $\text{prop} \leftarrow \max\{\text{prop}_q \mid (q, m_q, \text{prop}_q) \in M, q \in R^{r-1}\}$
 - 16 $m \leftarrow \text{prop}$
- 17 **else if** $(q, m_q, \text{prop}_q) \in M$ **with** $m_q \neq \perp$ **then**
 - 18 $\text{prop} \leftarrow m_q$
 - 19 $m \leftarrow \perp$
- 20 **else**
- 21 $m \leftarrow \perp$

Termination occurs at the latest when the two successive graphs G, G' of $\Diamond\text{STAR}(2)$ occur in rounds $r', r' + 1$: In round $r' + 1$, every process detects R , the root component of G and sets $\text{prop} = v$, where v is uniquely determined by R . Subsequently, any process sends and receives only messages $(*, v)$. \blacktriangleleft

7

Conclusions

We provided tight upper and lower bounds for the solvability of consensus under message adversaries that guarantee a stable root component only eventually and only for a short period of time: We showed that consensus is solvable if and only if each graph has exactly one root component and, eventually, there is a period of at least $D + 1$ consecutive rounds (with $D \leq n - 1$ denoting the number of rounds required by root members for broadcasting) where the root component remains the same. We also provided a matching consensus algorithm, along with its correctness proof. While this kind of short-lived periods of stability is useful from the perspective of assumption coverage in real systems and fast termination time, we also demonstrated that longer periods of stability allow the development of less complex algorithms.

Part of our current/future work is devoted to simulations of message adversaries, which also includes exploring their relation to failure detectors.

References

- 1 Y. Afek and E. Gafni. Asynchrony from synchrony. In D. Frey, M. Raynal, S. Sarkar, R. Shyamasundar, and P. Sinha, editors, *Distributed Computing and Networking*, volume 7730 of *Lecture Notes in Computer Science*, pages 225–239. Springer Berlin Heidelberg, 2013.
- 2 M. Biely, P. Robinson, and U. Schmid. Agreement in directed dynamic networks. In *Proceedings 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO'12)*, LNCS 7355, pages 73–84. Springer-Verlag, 2012.
- 3 M. Biely, P. Robinson, U. Schmid, M. Schwarz, and K. Winkler. Gracefully degrading consensus and k -set agreement in directed dynamic networks. In *Revised selected papers Third International Conference on Networked Systems (NETYS'15)*, Springer LNCS 9466, pages 109–124, Agadir, Morocco, 2015. Springer International Publishing.
- 4 M. Biely, U. Schmid, and B. Weiss. Synchronous consensus under hybrid process and link failures. *Theoretical Computer Science*, 412(40):5602 – 5630, 2011. <http://dx.doi.org/10.1016/j.tcs.2010.09.032>.
- 5 B. Charron-Bost, M. Függer, and T. Nowak. Approximate consensus in highly dynamic networks: The role of averaging algorithms. In M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann, editors, *Automata, Languages, and Programming*, volume 9135 of *Lecture Notes in Computer Science*, pages 528–539. Springer Berlin Heidelberg, 2015.
- 6 B. Charron-Bost and A. Schiper. The Heard-Of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, Apr. 2009.
- 7 É. Coulouma, E. Godard, and J. G. Peters. A characterization of oblivious message adversaries for which consensus is solvable. *Theor. Comput. Sci.*, 584:80–90, 2015.
- 8 M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, Apr. 1985.
- 9 W. Kiess and M. Mauve. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Netw.*, 5(3):324–339, Apr. 2007.
- 10 F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *STOC*, pages 513–522, 2010.
- 11 F. Kuhn, R. Oshman, and Y. Moses. Coordinated consensus in dynamic networks. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, PODC '11. ACM, 2011.
- 12 F. Legendre, T. Hossmann, F. Sutton, and B. Plattner. 30 years of wireless ad hoc networking research: What about humanitarian and disaster relief solutions? What are we still missing? In *International Conference on Wireless Technologies for Humanitarian Relief (ACWR 11)*, Amrita, India, 2011. IEEE.
- 13 M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 39–49, New York, NY, USA, 2004. ACM.
- 14 D. Pfleger and U. Schmid. A framework for connectivity monitoring in wireless sensor networks. In *Proceedings 10th International Conference on Sensor Technologies and Applications (SENSORCOMM'16)*, pages 40–48. IARIA, 2016. https://www.thinkmind.org/download.php?articleid=sensorcomm_2016_3_10_10013.
- 15 M. Raynal and J. Stainer. Synchrony weakened by message adversaries vs asynchrony restricted by failure detectors. In *Proceedings ACM Symposium on Principles of Distributed Computing (PODC'13)*, pages 166–175, 2013.
- 16 N. Santoro and P. Widmayer. Time is not a healer. In *Proc. 6th Annual Symposium on Theor. Aspects of Computer Science (STACS'89)*, LNCS 349, pages 304–313, Paderborn, Germany, Feb. 1989. Springer-Verlag.

- 17 U. Schmid, B. Weiss, and I. Keidar. Impossibility results and lower bounds for consensus under link failures. *SIAM Journal on Computing*, 38(5):1912–1951, 2009.
- 18 M. Schwarz, K. Winkler, and U. Schmid. Fast consensus under eventually stabilizing message adversaries. *arXiv:1508.008516*, August 2015. in Proc. ICDCN’16).
- 19 M. Schwarz, K. Winkler, and U. Schmid. Fast consensus under eventually stabilizing message adversaries. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, ICDCN ’16, pages 7:1–7:10, New York, NY, USA, 2016. ACM.
- 20 M. Schwarz, K. Winkler, U. Schmid, M. Biely, and P. Robinson. Brief announcement: Gracefully degrading consensus and k -set agreement under dynamic link failures. In *Proceedings of the 33th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC ’14, pages 341–343, New York, NY, USA, 2014. ACM.